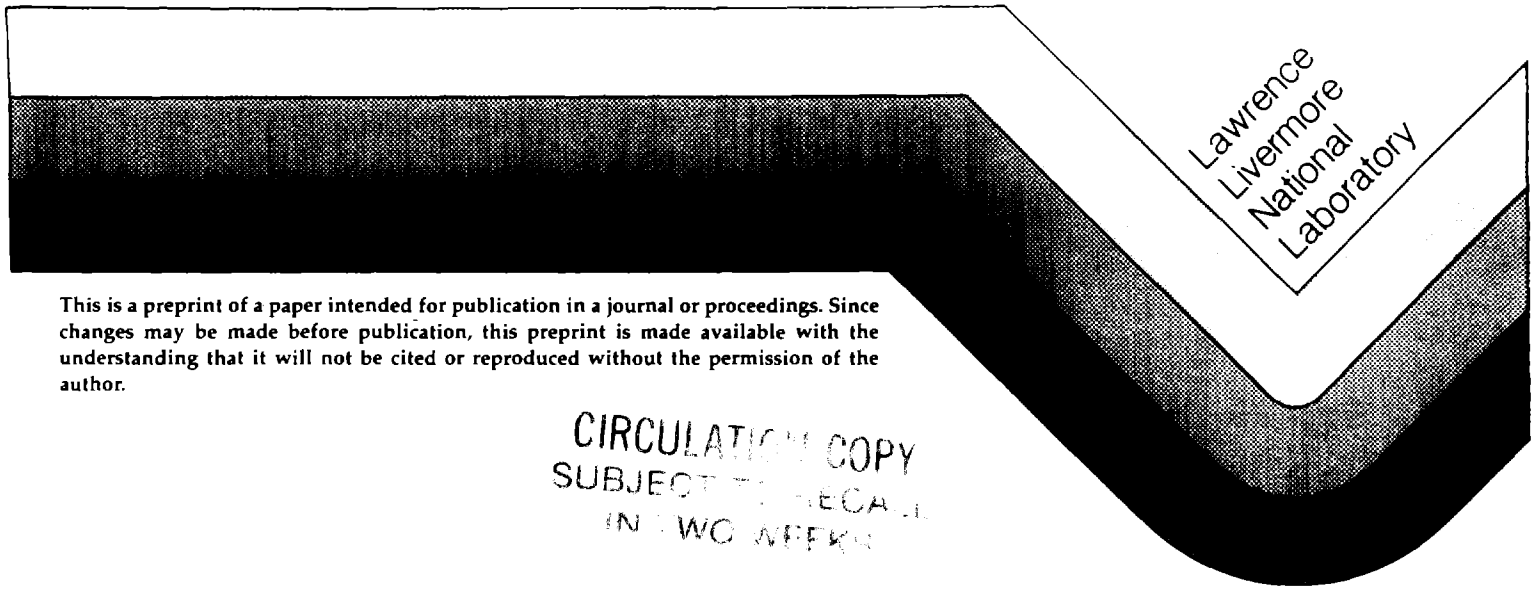
UCRL- 96080, Rev. 1
PREPRINT

LAYERING CENTRAL AUTHENTICATION ON EXISTING DISTRIBUTED SYSTEM INTERACTIVE SERVICES

Dan M. Nessett

This paper was prepared for submission to the 8th International Conference on Distributed Computing Systems sponsored by the IEEE Computer Society, San Jose, California, June 13-17, 1988.

October 1987



LAYERING CENTRAL AUTHENTICATION ON EXISTING DISTRIBUTED SYSTEM INTERACTIVE SERVICES

D. M. Nessett

Lawrence Livermore National Laboratory

Livermore, CA 94550

Abstract - Provision of interactive terminal service in a distributed system requires mechanisms to logon and logoff as well as to move textual data between the terminal host and remote host. Logon occurs in most distributed systems subsequent to the establishment of a terminal session by means of host specific logon procedures. However, in a distributed system of any size, this approach leads to security and password management problems. When the distributed system is centrally administered, these problems can be rectified through the use of a central authentication service that presents a common logon interface to the user for all distributed system hosts.

Normally, central authentication is provided by either initially designing it into a distributed system or supporting it through the modification of distributed system and host operating system software. As an alternative strategy, central authentication can be layered onto existing interactive terminal services. Research to assess the feasibility of this approach was carried out. The results demonstrate that layering can be used in certain circumstances to provide central authentication services. The research also determined what terminal service features are necessary so that central authentication is easily layered on existing interactive terminal services. Recommendations are made concerning how to structure terminal services in a distributed system to support an integrated central authentication service.

1. INTRODUCTION

Terminal access to hosts in a distributed system requires the provision of three basic services: 1) a user must be able to **logon** to and **logoff** from distributed system hosts, 2) the distributed system must move textual data between the terminal and the host, and 3) this data must be translated between the format understood by the terminal and that understood by the host. The structure of these services and of their supporting mechanisms is called the **interactive service architecture** of the distributed system.

Some interactive service architectures include two protocols that support these services, one called the **logon protocol** and the other the **virtual terminal protocol** [11]. The logon protocol allows the establishment of properly authenticated terminal sessions, while the virtual terminal protocol provides all remaining interactive services. However, in many cases an interactive service architecture provides only a virtual terminal protocol, relying on mechanisms within each host to support logon.

Interactive service architectures that rely soley on host mechanisms to achieve logon have the following inadequacies:

"Work performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under contract number W-7405-ENG-48."

- As the number of hosts with which a user interacts increases, the number of different passwords that he must remember also increases. Past experience shows that such situations encourage users to compromise security by writing down or otherwise physically recording these passwords on unprotected media (e.g., write them on a slip of paper taped to the side of their desk or place them in the memory of smart terminals so that they can be transmitted by pressing a function key), or by using passwords that are easily guessed.
- Each distributed system host is required to maintain its own protected password database. When a distributed system is constituted of different types of hosts, each type presents a different password management interface to the user (e.g., the number and type of characters that can be used in a password, the maximum lifetime of the password, whether the user or the system chooses new passwords). Dealing with multiple password management interfaces in a distributed system can lead to confusion about what passwords are legitimate, when they should be changed, and so forth.
- If it ever becomes necessary to revoke a user's access privileges to the distributed system, all hosts must be instructed to invalidate that user's identifier and password in their password database. This is both time consuming and operationally complex.

These problems can be met in a number of ways. One approach is to assign a single password to a user and then disseminate it to each host. This scheme has the following disadvantages:

- Updating a password requires changing it on more than one host. This, in turn, requires the use of a distributed replicated data base management mechanism that supports reliable data base update to ensure consistency. Support of such an update mechanism requires a significant investment in software design, implementation and maintanence.
- Storing the same password in more than one place raises the risk of password compromise [9]. This is especially significant if some systems are more secure than others.
- Not all hosts allow the same set of passwords to be used. Choosing a password from a common subset can lead to security problems if the number of passwords in the subset is small. Small password sets allow password search activity by an intruder to succeed with an unacceptably high probability.
- Some systems do not support the concept of a disseminated password. They are designed to choose passwords themselves and distribute them to the user.
- It is still necessary to contact all hosts to revoke a user's access privileges to the distributed system.

Another scheme is to trust hosts to guarantee a user's identity during logon. This approach is taken by the *rlogin* mechanism supported in UNIX BSD 4.2 and 4.3 [14]. Its disadvantages are:

- If one host is compromised, it potentially compromises all hosts that trust it. Once these hosts are compromised, they, in turn, potentially compromise all hosts that trust them. Unless the transitive closure of the trust relation is carefully controlled (which is unlikely from a practical point of view), compromise of one host in the distributed system compromises them all. More succinctly stated, such mechanisms violate the principle of mutual suspicion which is a fundamental criterion of distributed system security [15, 13].
- If one host can masquerade as another by using its internetwork or other host-level address (e.g., ethernet address), it can gain access to and potentially compromise other hosts in the distributed system. Thus,

one workstation in the hands of an intruder can compromise a distributed system by masquerading as other hosts and then taking advantage of the transitive closure of the trust relation.

- Again, it is necessary to contact all hosts to revoke a user's access privileges the distributed system.

Since neither of these approaches is satisfactory when the security of a distributed system is important, the provision of secure logon requires an alternative approach. One technique is to provide centralized facilities in the distributed system that are responsible for password management and to support access to these facilities through a logon protocol.

How a logon protocol operates depends on whether the distributed system is administered by one or many administrations. Distributed systems managed by multiple administrations are the most important case to consider when distributed system mechanisms are designed. However, with respect to interactive service provision, such distributed systems can be decomposed into parts administered by single administrations. If logon within these parts is secure, logon between them can be made secure by means of gateways and an interfacing logon protocol (see section 6). The research reported here describes how logon within distributed systems managed by a single administration can be made secure by means of a **central authentication** service. It was conducted as part of an effort investigating secure logon in distributed systems managed by multiple administrations.

In addition to password management and system administration issues, other security requirements affect distributed system logon services. For example, in a classified environment a terminal session may be established at one of many different security-levels. Determining whether a user is authorized to logon to a particular host at the level requested is an important decision during logon in these environments. A central authentication service must adequately support such decisions.

While some distributed systems are designed from inception to support central authentication [15, 16], most are not. Certain distributed systems have been enhanced to support central authentication by modifying their interactive service architecture [2, 3, 10]. Among other things this requires modifying the host software that supports distributed system logon. Such an approach is acceptible if host software is designed, implemented and maintained by the customer. However, when vendor supplied host software must be modified and maintained, this procedure can be expensive and is therefore unattractive.

It is also possible to enhance an existing interactive service architecture by layering central authentication onto existing interactive services, thereby eliminating costly changes to host operating system software. If successfully carried out, layering eliminates many design and maintenance problems that can occur when vendor supplied software is modified by customers.

Research exploring the layering approach was carried out. The research results indicate that layering is possible under the constraints and conditions described in section 4. More importantly, it was determined when layering central authentication on top of existing services is possible and when central authentication is best achieved by modifying the interactive service architecture. From the experience gained during the research effort, recommendations are made concerning how interactive service architectures should be designed so that central authentication can be effectively achieved.

The next section describes central authentication, discusses its advantages and disadvantages, and gives a brief history of its use. Section 3 describes the layering approach and section 4 discusses the research results. From the results, recommendations are made in section 5 about interactive service architecture design. Section 6 briefly

describes how central authentication can be used to support secure logon in distributed systems managed by multiple administrations.

2. CENTRAL AUTHENTICATION

Provision of central authentication service in singly administered distributed systems requires centralizing the password data bases of the distributed system hosts in a system called an **authenticator**. The role of the authenticator is to determine whether a user identifier and password provided in a logon request are properly matched, a process called **authentication**. If the user identifier and password are properly matched, both the logon request and the user are said to have been **authenticated**. If they do not match, an **authentication failure** has occured. Central authentication also requires an **authorization** function that determines whether a particular user is allowed to logon to a particular host. Authorization decisions can occur either in the authenticator or in the host.

Movement of information necessary for authentication and authorization is the responsibility of the logon protocol. A logon protocol supporting central authentication operates between three participants - the terminal host, authenticator and remote host. The software responsible for distributed system interactive services on these systems is implemented either by operating system kernel procedures, application processes or a combination of these (Fig. 1).

A terminal interacts with its terminal host by means of a **terminal process** that assembles/disassembles packets, interprets signals from the terminal such as break or attention key depression and handles any synchronization requirements such as matching a half-duplex terminal to full-duplex communications. In addition to logon protocol participation, the terminal process implements the terminal side of the virtual terminal protocol. The terminal process can be implemented by an operating system terminal driver, a system-level process, a user-level process or a combination of these.

The remote host participates in the logon and virtual terminal protocols through the combined activity of a number of entities. Hosts supporting directly connected as well as remote terminals normally use a **remote terminal driver** to support remote terminal sessions. The typical remote terminal driver assembles/disassembles packets, implements the host side of the virtual terminal protocol and presents a programming interface that allows processes running on the remote host to treat remote terminals as if they were directly connected. Before logon has occurred, the remote terminal driver passes terminal character data to a **logon-listener** that receives and validates logon requests. After logon, the remote terminal driver passes terminal character data to a **command language interpreter (CLI)** which parses and implements commands. Hosts that only support remote terminals may combine the remote terminal driver, logon-listener and CLI into a single computational entity (Fig. 2). For pedagogical reasons we assume that this is the case and speak of the CLI as the remote host entity responsible for logon and virtual terminal protocol activity.

It is the authenticator's role to service logon requests from the terminal process. Its responsibilities are described above.

Once a request has been authenticated, it must be determined whether the identified user is authorized to use resources managed by the specified CLI. This decision can be made either by the authenticator or by the individual CLIs. Providing the authorization function in the authenticator permits centralized control of user access to

distributed system resources. If CLIs perform authorization, resource access control is distributed to those who manage each CLI authorization data base. Whether authorization should be centralized or distributed depends on the management philosophy of the distributed system administration. If management responsibilities for distributed system elements are distributed, it is likely that authorization should be performed in the CLIs. If management responsibilities are centralized, then authorization by the authenticator is probably more suitable. For the remainder of section 2 it is assumed that authorization is performed by the CLIs. An example of authenticator-based authorization is given when the layering approach is described.

Central authentication can be achieved in a number of different ways. Central authentication design issues include the logon protocol architecture and how it is impacted by the virtual terminal protocol architecture, how source address guarentees are made in the distributed system, and how recovery from password compromise is accomplished. If the distributed system supports security-level labeling, it must be determined how to restrict logon by users to the range of levels they are authorized to use. Password management is an important area affected by central authentication. User identifiers must be added to the authenticator data base along with their associated passwords, it must be possible for users or an administrator to change passwords, and an administrator must be able to revoke the right of users to logon to distributed system hosts. These services may require an auxiliary protocol for password management or they may be provided by the logon protocol. In addition, mechanisms to protect password and terminal session data may affect the logon protocol architecture as well as terminal process, authenticator and CLI structure.

2.1 Around-the-Horn Logon Protocols

A logon protocol can be organized in one of two ways. An **around-the-horn** logon protocol requires the authenticator to forward an **authorization request** to the CLI after the successful authentication of a logon request (Fig. 3). The CLI determines whether the user identified in the request is authorized to use the host's resources and if so establishes a terminal session with the terminal process by means of the virtual terminal protocol. If the user is not authorized to use the host's resources, an authorization failure message is returned to the terminal process. Use of an around-the-horn logon protocol assumes that: 1) the virtual terminal protocol and its implementation allows the CLI to establish a terminal session, and 2) the terminal process can properly discriminate between authentic and bogus replies to a logon request (to prevent an intruder from masquerading as a CLI). These points are elaborated below.

2.2 Remote Procedure Call† (RPC) Logon Protocols

---

† The use of the phrase **remote procedure call** to describe this type of logon protocol may be controversial. It is used in the spirit which originally gave rise to the terminology. The terminal process makes a (remote) procedure call to the authenticator to obtain authentication service. It then makes a (remote) procedure call to the host to establish the terminal session.

A remote procedure call (RPC) logon protocol requires the authenticator to return authorization information to the terminal process after the successful authentication of a logon request (Fig. 4). The terminal process then contacts the CLI, providing this information as proof of authentication. The exact nature of the authorization information depends on the relationship of trust between the terminal process and the CLI. If the terminal process is trusted by all CLIs in the distributed system to guarentee that authentication has taken place (for example, when terminal processes are implemented on trusted tamper-proof terminal multiplexors), the authorization information might consist of the user identifier and an indication that the user has been properly authenticated [2]. The terminal process sends this indication to the CLI in a virtual-terminal-protocol terminal-session-establishment request. The CLI accepts the request if the user is authorized to use the host's resources. Otherwise, the CLI rejects the request. Secure operation requires the CLI to ensure that the terminal-session-establishment requests it receives come from a trusted terminal process and not from some intruder. This can be accomplished either by the provision of a trusted communications subsystem that guarentees source addresses or by means of message authentication based on encryption techniques [8].

If CLIs don't trust terminal processes to guarentee that authentication has taken place, more substantial authorization information must be provided by the authenticator. For example, in the layering approach described below the authenticator returns a secondary user identifier and password that allows the terminal process to logon to the host identified in the logon request. The secondary user identifier and password are submitted to the CLI in a logon request sent over a virtual-terminal-protocol terminal session established after the receipt of the authorization information (Fig. 5).

2.3 A Comparison of Around-The-Horn and RPC Logon Protocols

One advantage that an RPC logon protocol enjoys over an around-the-horn logon protocol is that under its rules of operation the terminal process, rather than the CLI, establishes the terminal session. This is the only mode of operation supported by most virtual terminal protocols or at least by their implementations. The terminal process still must be able to discriminate between authentic and bogus replies to its logon request; although, since these replies always come from the authenticator, reply validation can be accomplished by guaranteeing that the source address of the logon reply message is correct.

On the other hand, an RPC logon protocol employing reusable authorization information such as secondary user identifiers and passwords has the disadvantage that such information passes through the terminal process. If a terminal process is compromised, certain complications arise in restoring secure operations after compromise detection.

When a security compromise is detected, around-the-horn logon protocols allow the recovery of secure operations by changing all primary passwords that passed through the compromised terminal process. RPC logon protocols that rely on reusable authorization information require more elaborate and costly recovery procedures. Since the authorization information returned to the terminal process can be reused, an intruder can retain and use it after the primary passwords have been changed. In this case, compromise recovery requires changing all primary passwords and all reusable authorization information that passed through the compromised terminal process. This is a much more expensive and elaborate procedure than changing the authenticator's password data base since all involved CLIs

must be contacted. Designing an RPC logon protocol without universally trusted terminal processes that doesn't have this disadvantage requires that authorization information be valid for a single terminal session. This is not easily achieved.

Note that while a compromise remains undetected, neither type of logon protocol offers security advantages over the other. In both schemes primary user identifier and password data pass through the terminal process. This data allows an intruder who has compromised the terminal process to masquerade as the identified user.

2.4 Advantages and Disadvantages of Central Authentication

Some advantages associated with central authentication have already been pointed out. It requires the user to remember no more than one password and it presents him with a single password management interface. In addition, central authentication allows a centralized administrative authority quickly to revoke access to distributed system hosts by a particular user. If authorization is performed by the authenticator, even more refined control is possible. An administrator can specify and quickly change the set of hosts accessible by a user and in a classified environment can control the security-levels at which a user is allowed to operate on those hosts.

There are also disadvantages of central authentication. If no single administrative authority controls the distributed system, each must trust the one that controls the authenticator. In an atmosphere of mutual suspicion, this may not be possible. In that case an authentication approach is necessary that allows each administrative authority to trust only those other authorities it chooses and only to the degree it desires [12].

Secondly, the authenticator becomes a central point of failure in the logon process. Trying to eliminate this disadvantage by using multiple authenticators that are geographically distant from each other introduces problems of distributed authenticator data base management, selecting an authenticator to which to send logon requests and maintaining convergent authenticator software. However, it is possible to implement the authenticator on closely coupled redundant processors and to use redundant communications channels to ensure that down time is acceptably small.

Thirdly, central authentication requires the participation of the terminal processes and CLIs in the distributed system logon protocol. If these entities were not designed with such participation in mind, retrofitting them may require significant modification to their internal structure as well as to the structure of the existing interactive service architecture.

It has already been mentioned that an around-the-horn logon protocol requires the CLI to open a terminal session to the terminal process. If the virtual terminal protocol does not support terminal session establishment by a CLI, either it must be modified to do so (which requires modification of all of its implementations, a costly procedure), an out-of-band signal must be given to the terminal process so that it can establish the terminal session (which requires modifications to the terminal process and CLI) or an RPC logon protocol must be used, introducing the disadvantages described above.

Modifying existing terminal process and CLI programs may be undesirable because of the costs associated with the design, implementation, promulgation and maintenance of these changes. This is especially true in distributed systems with hosts from many different manufacturers, many of which may not be willing to provide central authentication support, forcing customer modifications to manufacturer software. Layering central authentication on

top of an existing interactive service architecture also requires design, implementation and maintenance efforts that are costly. The advantages and disadvantages of layering are discussed in Section 4.

2.5 History of Central Authentication

To the author's best knowledge, the first distributed system using central authentication is the Lawrence Livermore National Laboratory Octopus network [2, 3]. It uses an RPC logon protocol with reliance on trusted terminal multiplexors. An around-the-horn logon protocol was developed for the ICN network when central authentication was established at the Los Alamos National Laboratory. Kent, et al. [10] describe a central authentication service using multiple authenticators that controls terminal access to the Defense Data Network. Hosts also may use this service to authenticate their own logon requests.

The LINCS distributed operating system being implemented at Lawrence Livermore National Laboratory uses an around-the-horn logon protocol in concert with capability-based resource access control to establish a terminal session with a CLI [4, 16]. A LINCS CLI can maintain many different contexts for the same user as well as contexts for different users and there may be many CLIs running on a single host.

While technically not a central authentication scheme as defined in this paper, Israel and Lindon describe a context establishment mechanism for the Xerox internetwork with features similar to central authentication [7]. Their approach is unique in that logon does not connect a terminal to a host, but rather brings a user context to a personal workstation. User passwords are stored in a distributed data base system called the Clearinghouse, and logon requests are directed to it from a workstation. Upon authentication by the Clearinghouse, one of two things happen. If the user context is stored on the workstation at which the user is located, the user is allowed to access it. If the context is not on that workstation, it is retrieved either from the internetwork file system or some other workstation and moved to the workstation at which the user sits.

3. THE LAYERING APPROACH

3.1 Research Objectives

Research was carried out to explore the layering approach to central authentication. The main research objectives were to identify factors that are important when enhancing an interactive service architecture to support central authentication; to determine whether this approach is satisfactory; and if not, to identify what changes to an interactive service architecture are necessary to achieve an acceptable central authentication service. DECNET was chosen as the target distributed system architecture and VMS as the host operating system over which the layering software was placed.

A collateral research objective was to design the central authentication service to allow terminal sessions to be established at one of a number of different security-levels. Neither DECNET nor VMS support security-level labeling. This necessitated layering the labeling service on top of existing VMS and DECNET services. As explained in section 4.2, a number of difficulties were encountered in the pursuit of this objective.

3.2 The Virtual Terminal Protocol

The DECNET interactive service architecture supports two virtual terminal protocols, LAT and CTERM. The LAT protocol provides communications between terminal servers (i.e., terminal multiplexers) and DECNET hosts over an ethernet. Since LAT does not support store-and-forward services through DECNET nodes, while CTERM does, it was decided to layer the central authentication service over CTERM.

CTERM is based on a remote system-call service paradigm similar in philosophy to remote procedure call. The advantage of the remote system-call approach is that programs see the same terminal interface whether terminals are local or remote.

The disadvantage of this approach is the restrictive homogeneity required for it to work. Since system-calls are passed between the remote and terminal hosts, both systems must have the same or very similar operating system interfaces, at least with respect to terminal services. Such an environment is unlikely to exist unless all systems are from the same manufacturer or from manufacturers making compatible equipment. While a subset of the CTERM protocol is implemented on an operating system of significantly different architecture than VMS [6], this required greatly restricting the protocol services available to processes.

3.3 Layering Central Authentication Over DECNET

Layering central authentication over DECNET requires authenticator, terminal process and logon protocol support. In keeping with the layering approach, provision of these services is accomplished without modification to the host operating system.

3.3.1 Security-Level Labeling. The logon protocol software allows a terminal session to be established at one of several possible security-levels. The authenticator parses the requested terminal session security-level from the logon-line, using it in a number of tests to determine whether to establish a terminal session (see section 3.3.2). Since neither VMS nor DECNET supports the labeling of resources with a security-level, the logon protocol rather than the network-level protocol is used to communicate security-level information between the terminal process and the authenticator.

Similarly, the CTERM protocol does not support security-level labeling. Thus, no security-level is communicated between the terminal process and the remote host when a terminal session is established. Operational constraints are necessary to ensure that terminal session data is properly secured. They are discussed in section 4.2.

3.3.2 The Authenticator. The authenticator allows a fine granularity of control over logon requests. To provide centralized control of network resources, the authenticator supports both authentication and authorization functions. Upon receipt of a logon request, the authenticator:

- determines whether the user identifier and password given in the logon-line are properly matched,
- determines whether the host is allowed to participate in terminal sessions at the security-level specified in the logon-line (i.e., the security-level being requested for the terminal session),
- determines whether the terminal is allowed to participate in terminal sessions at the requested security-level, and
- determines whether the user is authorized to run on the identified host at the requested security-level.

If these tests succeed, the authenticator retrieves a secondary user identifier and password valid on the remote host. This is communicated to the terminal process in the manner described below.

3.3.3 The Logon Protocol. Since the CTERM protocol allows only terminal processes, not hosts to initiate terminal sessions, an RPC logon protocol was employed. The following sequence of events occur during logon:

- The CTERM program is executed by the terminal process. This can occur either by requiring the user to execute the program from the command language interpreter, or by using the automatic logon and captured account features of VMS to bind a particular terminal to the execution of the program.
- The terminal process prompts the user for a machine identifier, user identifier, security-level and password. These are sent to the authenticator in a logon request. If the logon request is valid, the authenticator returns a secondary user identifier and password to the terminal process. If the logon request is invalid, an error message is returned to the terminal process which is displayed to the user.
- If the logon request is valid, the terminal process uses the secondary user identifier and password to establish a terminal session with the target machine.

4. RESULTS

Layering central authentication over existing interactive services turned out to be more complicated and time consuming than originally expected. The research required approximately two man-years to complete, most of the effort going into the CTERM protocol implementation. While the layering objective was successfully accomplished, upgrading the layering software and then placing it into production would result in a significant maintenance effort, especially tracking changes to the CTERM protocol. For this reason it is unclear that the layering approach ultimately requires less customer support than would be required to embed a central authentication service into host operating system software.

The authenticator required the least implementation effort even though it was written from scratch. Implementing the terminal process turned out to be very time consuming because of the complexity inherent in the CTERM protocol, while the logon protocol was fairly easy to implement.

4.1 The Logon Protocol

During the logon protocol design, a decision was required whether to use the around-the-horn or RPC technique. An around-the-horn logon protocol was considered, but rejected for the following reasons:

- CTERM's asymmetry (allowing only terminal processes to open a terminal session) makes it difficult to implement an around-the-horn logon protocol (see section 2.4).
- Implementing a seperate virtual terminal protocol for DECNET that provides the necessary features for an around-the-horn logon protocol would be a major undertaking. In addition to the costs associated with implementing a new virtual terminal protocol, layering it on top of the VMS operating system would be difficult.
- An around-the-horn logon protocol requires communications between the authenticator and the CLI as well as between the authenticator and the terminal process. An RPC logon protocol only requires

communications between the authenticator and terminal process. Thus, an RPC logon protocol normally is simpler and quicker to implement than an around-the-horn logon protocol.

However, because of the way CTERM works, the RPC logon protocol has a number of disadvantages:

- Secondary user identifier and password data sent to complete the logon process must pass through a terminal process that is vulnerable to intruder access (especially if the terminal process runs on a workstation). Thus, it can be retained and used at a later date, creating the compromise recovery problems described section 2.3.
- To ensure logon control is maintained by the authenticator, secondary password data must be changed whenever a user's access rights to a host change. This requires a supplementary protocol by which the authenticator requests secondary password data update by a remote host. The requirement for such a protocol somewhat offsets the RPC logon protocol advantage of not requiring communications between the authenticator and hosts for logon purposes.
- To accomplish logon, the terminal process must maintain logon-phase state information indicating whether the user identifier or password has been sent and whether logon has completed successfully. To determine changes in the logon-phase, the program must examine character strings embedded in the terminal session data sent by the remote host that indicate the success or failure of the logon process. If such strings change over versions of the host operating system, modifications to the program are necessary to ensure correct operation.

The experience of adding a logon protocol to DECNET leads to the recommendations for interactive service architecture design given in section 5.

4.2 Terminal Session Security-Level Labeling

Since VMS does not support security-level labeling, each host can handle information at only one security-level. All resources managed by a host are implicitly labeled at that level. In addition, since DECNET network-level messages do not carry a security-level, intermediate hosts are unaware of the security-level associated with the data they store-and-forward. These two characteritics create a number of problems related to the support of labeled terminal sessions.

When a logon request is received by the authenticator, the security-level in the request is compared with the level of the remote host (which is kept in an internal authenticator table) to ensure their equality. This guarentees that terminal sessions to that host always operate at its level. However, the terminal process also runs on a host that can process information at only one level. This implies that the authenticator must ensure that the terminal and remote host operate at the same security-level.

Strict adherence to the single-level host requirement leads to severe operational inconvenience. In particular, terminal sessions established from a particular terminal can run at exactly one level and therefore must only connect the terminal to hosts running at that level. If a user needs to access hosts of different security-levels, the use of seperate terminals connected to different terminal hosts is necessary, one terminal for each level. Obviously, this is greatly inconvenient, especially if a user works from his office.

A number of solutions to this problem are possible. Security-level labeling could be added to VMS and DECNET. This approach is cost prohibitive for VMS and DECNET customers.

Terminals could be connected to user activated line switches that physically connect/disconnect them to/from different terminal hosts. When a user wishes to operate at a particular security-level, he selects the appropriate terminal line that connects his terminal to a host running at that level. While this approach allows a user to contact hosts at different security-levels by means of a single terminal, it requires that offices or other areas containing terminals are wired with multiple terminal lines. This is both costly and contrary to modern trends that minimize physical plant wiring. Furthermore, for each security-level that a user might use, a seperate terminal line from a host operating at that level must be brought to the terminal. This may not be physically possible, independent of wiring costs.

Another approach is to use certain hosts as trusted terminal multiplexors (fig. 6). Since VMS, like most commercially available operating systems, has a number of security vulnerabilities, no programs written by users other than trusted system programmers should be allowed to run on the multiplexors. This requires that all logon requests to the multiplexors (other than from a directly connected operator console) be denied. All terminals supporting the establishment of terminal sessions at more than one level must be connected to one of these trusted terminal multiplexors. To ensure that primary password data does not travel through vulnerable hosts, each trusted terminal multiplexor must be directly connected to the authenticator. In addition, the CTERM program must be trusted to separate terminal session data so that information at different security-levels is not mixed and DECNET must be trusted to seperate transport-level connections.

The trusted terminal multiplexor approach also can be used to overcome the lack of security-level labeling in the DECNET network-level protocol. Hosts operating at a particular level can be segregated into sub-networks by connecting them only to other hosts operating at that level or to one or more multiplexors. In this way data traveling through general purpose hosts is always at the same security-level. Data traveling through the multiplexors is of multiple levels, but they only run the CTERM program which is assumed to properly segregate terminal session data.

While the trusted terminal multiplexor approach is viable, it is only partially satisfactory because:

- Secure operation involves trusting the CTERM program. While written to properly separate terminal sessions, critical review of its structure is necessary to develop confidence that it achieves this objective.
- Secure operation also involves trusting VMS not to mix data from separate DECNET transport-level connections. While this is probably a safe assumption, a detailed analysis of the relevant parts of VMS and DECNET implementations is necessary to develop confidence that this is true.
- The trusted terminal multiplexor approach imposes network topology constraints that may not be possible or may be very costly or inconvenient in some operational environments. Requiring the connection to trusted terminal multiplexors of all terminals that support multiple levels may impose costly plant wiring or equipment upgrades of the network. Segregating hosts into sub-networks operating at a single security-level can be costly due to the increased network connectivity required (nearby hosts may not be allowed to store-and-forward traffic through each other), the increased failure sensitivity of the star-like network topology, and the seperation of host functions (i.e., multiplexor hosts and general purpose hosts) that may underutilize distributed system resources.

Distributed system and operating system architects must become aware that security-level labeling is an important design issue. Retrofitting operating systems and distributed system protocols to support security-level labeling is non-trivial. Supporting applications that require security-level labeling with operating systems and protocols lacking this service leads to clumsy and inconvenient operational constraints such as those described above. Architects who desire that their systems operate in an environment in which security-level labeling is important should include this security service in their initial designs.

5. RECOMMENDATIONS

5.1 Requirements for Layering Central Authentication on an Existing Interactive Service Architecture

This research identified general characteristics of an interactive service architecture that are necessary for central authentication layering to be successful. Specifically:

- It must be possible to implement the virtual terminal protocol or at least the terminal session establishment phase of the protocol in a user process. If terminal session establishment is carried out solely by operating system code, protocol implementation changes necessary to contact the authenticator cannot be made without modifying the underlying operating system. Such changes violate the layering objective.
- If primary/secondary user identifiers and passwords are used, host password management interfaces must allow externally controlled password update. If the remote host updates passwords, the authenticator cannot maintain the proper mapping between primary and secondary data, since notification of password update returns to the terminal host rather than to the authenticator. Allowing the terminal host to forward the new password to the authenticator would introduce a denial of service hazard whereby a compromised terminal host could completely change the authenticator's password data base (i.e., by sending a bogus update message for each user in the distributed system).
- For general central authentication service, the virtual terminal protocol cannot be tied to a particular network-level mechanism. This type of limitation is present in the LAT protocol.

5.2 Recommended Interactive Service Architecture Features for Embedding Central Authentication

While the feasibility of layering central authentication on an existing interactive service architecture was demonstrated, the research results indicate that central authentication is more efficiently, conveniently and maintainably supported when it is an embedded service. To support such service, the architecture should possess certain features. In particular:

- The use of an RPC logon protocol is preferable when the virtual terminal protocol does not support terminal session establishment from the host/CLI side. Since RPC logon protocols based on reusable authorization information are susceptible to compromise control problems and since RPC logon protocols that rely on

trusted terminal multiplexors are impractical in many environments, virtual terminal protocols should support host/CLI side terminal session establishment. This allows the use of an around-the-horn logon protocol which has superior password compromise recovery properties. However, use of an around-the-horn logon protocol requires a mechanism whereby terminal processes can discriminate between valid and bogus responses to their logon requests. For example, the LINCS stream number mechanism [5] provides the necessary service to accomplish this.

- If an around-the-horn logon protocol is used, some mechanism must guarantee that authorization requests come from the authenticator. If the underlying connectivity of the distributed system is properly protected, source address guarantees are possible. If such protection is not available, message authentication based on encryption techniques [8] can be employed.
- The provision of a central authentication service is more convenient if CLIs are explicitly designed for such operation. CLIs that are expected to operate in environments that support central authentication, as well as in environments that do not, should be designed to separate their implementation into a module that handles logon and one that performs other services. Clean separation of the logon and command language interpretation functions allows different styles of logon to be supported by replacing the logon module. Module replacement by customers should be possible so that they can employ authenticator and logon protocols suitable to their needs.
- If a distributed system is to operate in an environment in which security-levels are required, its protocols and host operating systems should support security-level labeling. Layering a labeling service on top of existing distributed system services is difficult and leads to unsatisfactory operational requirements such as the use of trusted terminal multiplexors.

In addition to these requirements, those designing interactive service protocols should consider combining logon and virtual terminal protocols. Logon is viewed most effectively as the secure establishment of a terminal session; while logoff is best viewed as terminal session closing. Separating terminal session opening activity from other terminal session support services by means of a logon protocol obfuscates and complicates interactive service provision. New research and standards activity is required to develop and promulgate second-generation interactive service protocols that provide logon as well as virtual terminal services.

6. FUTURE DIRECTIONS

This paper describes and analyses one approach to the secure logon problem in distributed systems managed by a single authority. However, the research presented here was conducted as part of an investigation into logon support in multiply administered distributed systems. A brief description of the relationship between the two problems is now given.

Solving interactive service security and password management problems in a multiply administered distributed system is possible by partitioning it into singly administered parts called **authentication domains** [15]. Within an authentication domain, central authentication is used. Logon between authentication domains is possible by using **inter-authentication-domain (IAD) gateways** and an **IAD logon protocol** in the way now described.

Inter-authentication-domain logon uses the central authentication services of three authentication domains: 1) the authentication domain in which the terminal is located (the terminal AD), 2) the authentication domain in which the user's password is stored (the authenticator AD), and 3) the authentication domain in which the remote host is located (the host AD). These authentication domains need not be distinct. For example, the terminal AD and host AD may be identical.

Logon between authentication domains utilizes an around-the-horn logon protocol and proceeds as follows. The logon protocol of the terminal AD is used to forward a logon request from a terminal to its IAD gateway. The IAD gateway determines the location of the authenticator AD from information embedded in the user identifier given in the logon line. The gateway then forwards an IAD logon request to the authenticator AD. The IAD gateway of the authenticator AD receives the logon request and uses its logon protocol to authenticate the request. The gateway then determines the host AD from information embedded in the host identifier given in the logon line. The gateway forwards an IAD authorization request to the host AD. The IAD gateway of the host AD uses its logon protocol to logon to the remote host. It then returns an IAD logon response to the terminal AD gateway. The terminal session established by the logon protocol begins at the terminal, continues through the terminal AD and host AD gateways, and ends at the remote host. A full specification of the IAD logon protocol is contained in [12].

7. ACKNOWLEDGMENT

The author would like to express his gratitide to Brian Cabral, John Fletcher, Tony Genovese, Alex Phillips and Dave Wiltzius for their helpful comments and aid in carrying out the central authentication layering effort. Special thanks is given to John Fletcher for his help in understanding the central authentication problem. Thanks is given also to Dick Watson whose useful comments greatly improved the paper's organization.

8. REFERENCES

[1] J. Davidson, et al., "The ARPANET TELNET protocol: its purpose, principles, implementation, and impact on host operating system design," Proc. of the 5th Data Communications Symposium, Sept., 1977, pp. 4.10-4.18.

[2] J. G. Fletcher, "Combination checker software," Lawrence Livermore National Laboratory internal working document, Aug., 1972.

[3] J. G. Fletcher, "How the network works," Lawrence Livermore National Laboratory Report UCID-30072, Oct., 1972.

[4] J. G. Fletcher, "LINCS interactive terminal protocols," Lawrence Livermore National Laboratory internal working document, 11 July, 1983.

[5] John G. Fletcher, "Stream numbers," Lawrence Livermore National Laboratory internal working document, August 9, 1985.

[6] John Forecast, James L. Jackson and Jeffrey A. Schriesheim, "The DECNET-ULTRIX software," Digital Technical Journal, No. 3, Sept. 1986, pp. 100-107.

[7] J. E. Israel and T. A. Lindon, "Authentication in office systems," ACM Trans. on Office Systems, Vol. 1, No. 3, July, 1983, pp. 193-210.

[8] R. R. Jueneman, S. M. Matyas and C. H. Meyer, "Message authentication," IEEE Communications Magazine, Sept., 1985, pp. 29-40.

[9] S. T. Kent, "Security in computer networks," Chapter 7 in Protocols and Techniques for Data Communication Networks, ed., F. F. Kuo, Prentice-Hall, N.J., 1981.

[10] S. T. Kent, P. J. Sevcik and J. G. Herman, "Personal authentication system for access control to the defense data network," Proc. of EASCON 82, Washington, D.C., pp. 61-75.

[11] F. Magnee, A. Endrizzi, and J. Day, "A survey of terminal protocols," Computer Networks, Vol. 3, No. 5, Nov., 1979, pp. 299-314.

[12] D. M. Nessett, "The inter-authentication-domain (IAD) logon protocol (preliminary specification and implementation guide), Lawrence Livermore National Laboratory Report UCID-30207 (Rev. 1), Nov. 12, 1985.

[13] D. M. Nessett, "Factors affecting distributed system security, Proc. 1986 IEEE Symposium on Security and Privacy, Oakland, CA, April, 1986, pp. 204-222, also appears in IEEE Trans. on Software Engineering, Feb., 1987, pp. 233-248.

[14] J. S. Quarterman, A. Silberschatz, and J. L. Peterson, "4.2 BSD and 4.3 BSD as examples of the UNIX system," ACM Computing Surveys, Vol. 17, No. 4, Dec. 1985, pp. 379-418.

[15] R. W. Watson and J. G. Fletcher, "An architecture for support of network operating systems services," Computer Networks, Vol. 4, No. 1, Feb., 1980, pp. 33-49.

[16] R. W. Watson, "Requirements and overview of the LINCS distributed operating system architecture," Proc. 13th CRAY Users Group Meeting, Paris, France, April 25-27, 1984. Also available as Lawrence Livermore National Laboratory Report UCRL-90906.

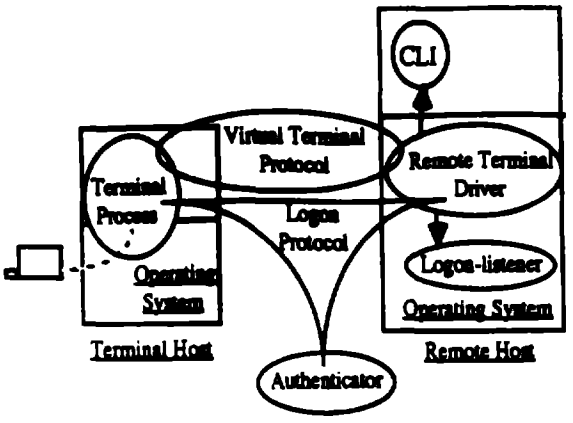


Fig. 1 - Remote logon to hosts supporting both local and remote terminals.

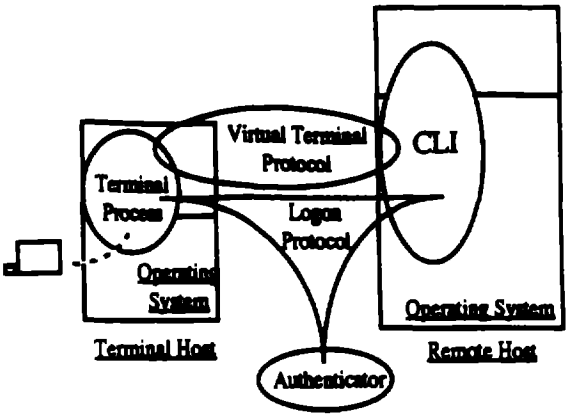


Fig. 2 - Remote logon to hosts supporting only remote terminals.

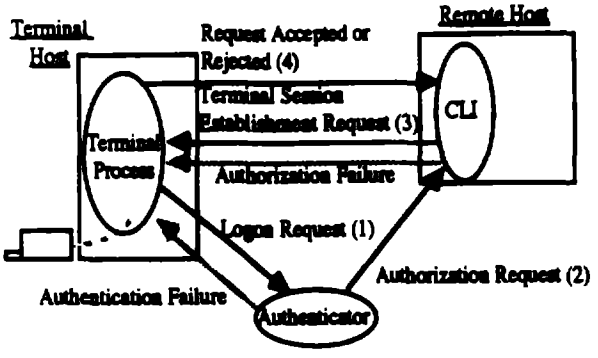


Fig. 3 - Around-the-horn logon protocol.

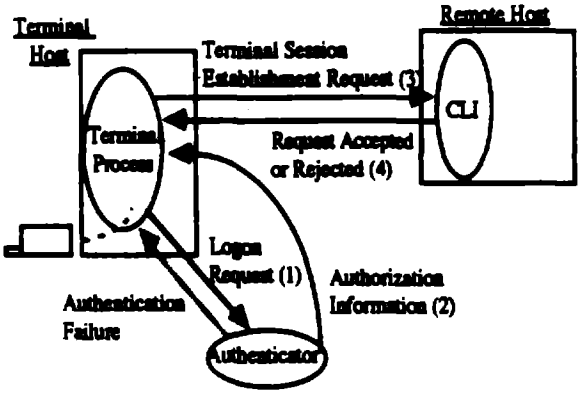


Fig. 4 - RPC logon protocol

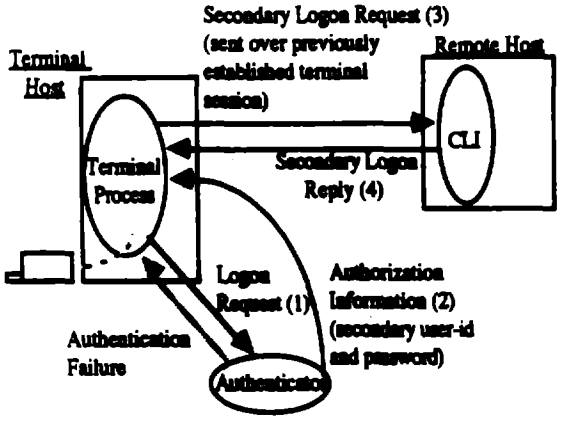


Fig. 5 - RPC logon protocol utilizing secondary user-id and password

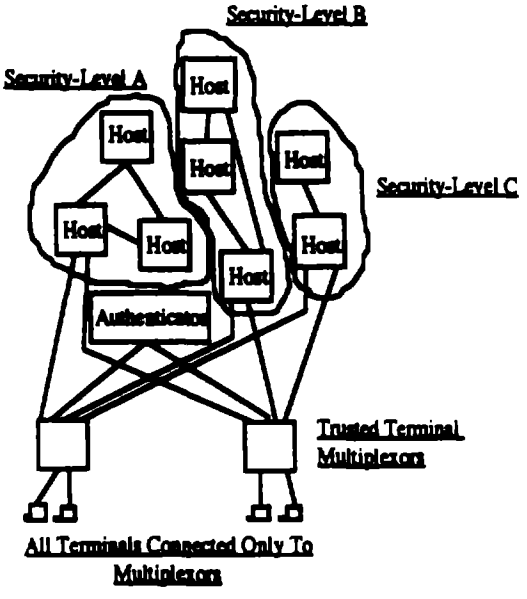


Fig. 6 - Using trusted terminal multiplexors to support security-level labeling